May 2019, IPT Course
Java Web Debelopment

Remote Method Invocation (RMI)

Trayan Iliev

tiliev@iproduct.org
http://iproduct.org

Where to Find the Code?

Intarmediate Java Programming projects and examples are available @ GitHub:

https://github.com/iproduct/course-java-web-development

Agenda for This Session

- RMI architecture
- Dynamic code loading
- Remote Interfaces, objects, and methods
- Using SecurityManager, security permissions and policies for accessing remote code
- Exporting the remote object using UnicastRemoteObject class
- Using RMI registry
- Remote exception handling
- Building sample RMI client-server application – ComputeEngine

Software Architecture

Fundamental organization of a system implemented through its components, the relationships between them and with the environment, as well as the principles guiding their design and evolution.

[ANSI / IEEE]

A set of important decisions about organization of a software system, the choice of structural elements and their interfaces, by which the system is composed, together with their behavior, as specified by collaborations between these structural and behavioural elements in progressively larger subsystems, and architectural styles guiding this organization.

[Booch, Rumbaugh, Jacobson, Unified Process]

Software Architecture

A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture.

A software architecture is defined by a configuration of architectural elements—components, connectors, and data—constrained in their relationships in order to achieve a desired set of architectural properties.

[Roy Fielding, Architectural Styles and the Design of Network-based Software Architectures]

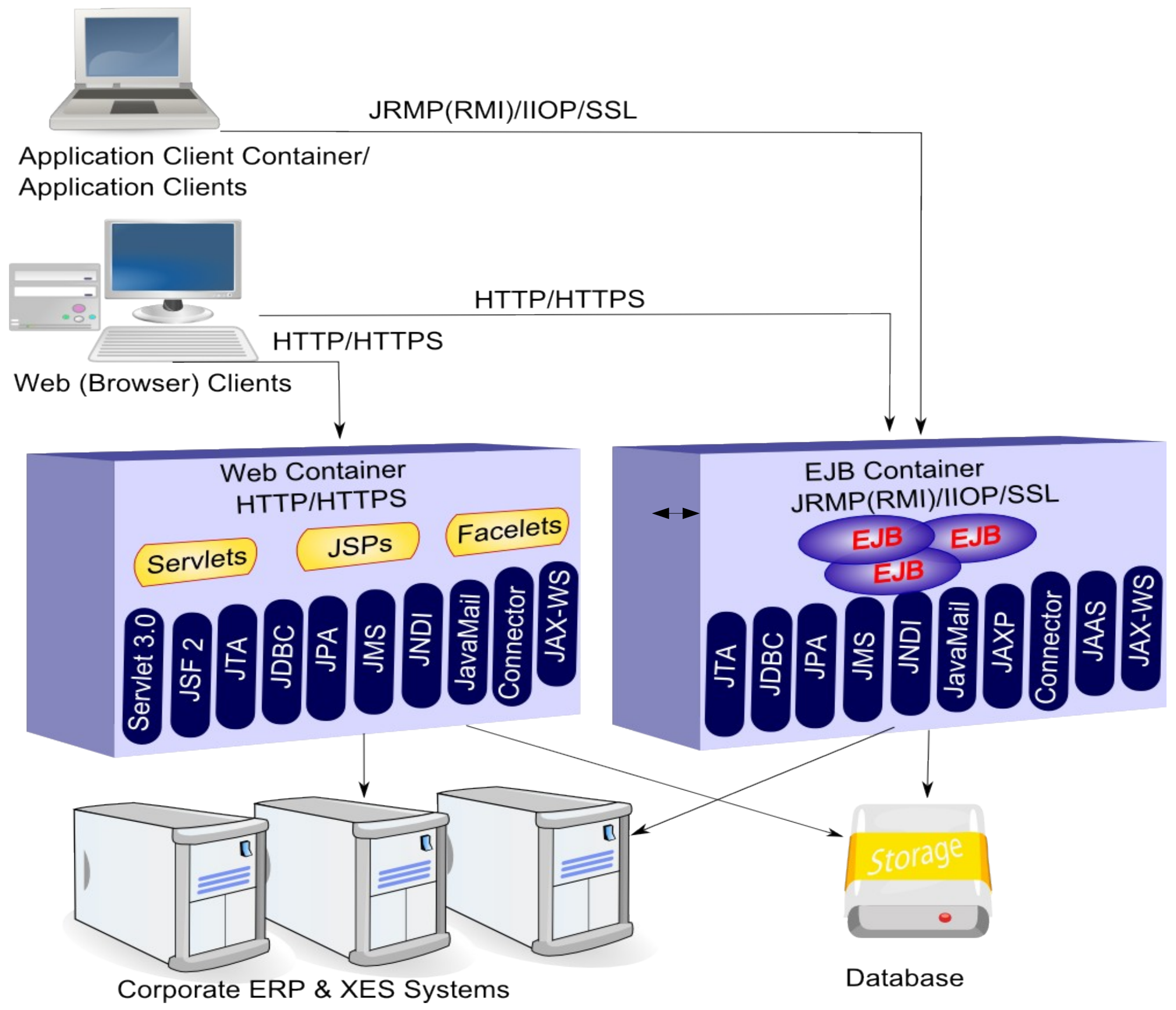
Views of the Architectural Model

- Functional/logic view
- Code/module view
- Development/structural view
- Concurrency/process/thread view
- Physical/deployment view
- User action/feedback view
- Data view

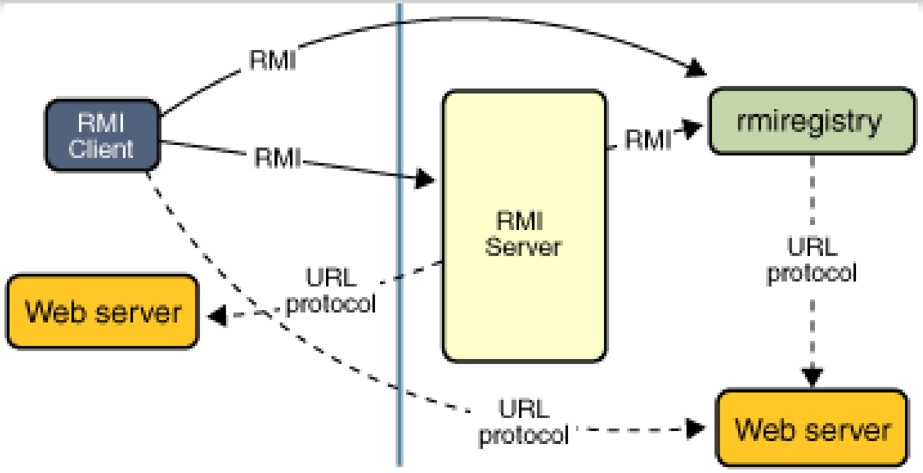
Architectural Styles

- Blackboard
- Client-server (2-tier, 3-tier, n-tier, cloud computing)
- Component-based
- Event-driven (or implicit invocation)
- Layered (or multilayered architecture)
- Microservices architecture
- Monolithic application
- Peer-to-peer (P2P)
- Pipes and filters
- Plug-ins
- Reactive architecture
- Service-oriented / Representational state transfer (REST)
- Rule-based

JavaEE Architecture



RMI Architecture

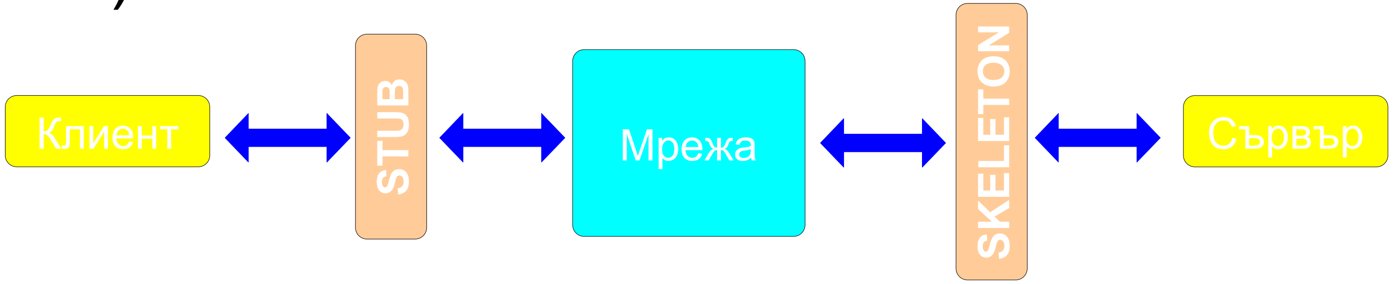


Distributed object applications need to do the following:

- Locate remote objects.
- Communicate with remote objects.
- Load class definitions for objects that are passed around.

Основни характеристики на RMI

- RMI = Remote Method Invocation
- RMI – реализация на Remote Procedure Calls (RPC) на езика Java



- Java клиент към Java сървър – сравнение с Common Object Requesting Broker Architecture (CORBA)

  Пример – IDL дефиниция: interface getPrice { float calculate_price (in float amount); }

Основни стъпки при реализация

- Дефиниране на интерфейс с основните бизнес методи, които ще бъдат викани отдалечено
- Реализиране на сървърен клас, който имплементира отдалечения интерфейс
- Регистриране на отдалечения обект
- ~~Създаване на Stub чрез RMI compiler – rmic~~
- Откриване (lookup) на отдалечения обект от клиента
- Извикване на отдалечените методи ит клиента
- Работа с SequrityManager и дефиниране на политики за сигурност – опции при стартиране на Java VM

Дефиниране на отдалечения интерфейс

- Интерфейсът трябва да бъде публичен
- Интерфейсът трябва да разширява java.rmi.Remote
- Интерфейсът трябва да декларира:

  throws java.rmi.RemoteException

- Отдалеченият обект достъпен от страна на клиента трябва да бъде деклариран от тип интерфейса, а не реализиращия го клас

Откриване и използване на отдалечения обект от клиента

При клиента:

```
    ProductController pc =
        (ProductController) Naming.lookup(
                        “//localhost:1099/ProductController”);
    List<Product> lp = pc.getProducts();
```

или

```
    Registry registry =
            LocateRegistry.getRegistry(“localhost:1099”);
    ProductController stub =
       (ProductController)
registry.lookup("ProductController");
```

Реализиране на отдалечения интерфейс и регистрация в RMI Registry

- Реализиращият отдалечения интерфейс клас трябва да наследява java.rmi.server.UnicastRemoteObject
- Необходимо е кодът който създава и регистрира инстанция на класа в RMI Registry да се стартира със SequrityManager - java.rmi.RMISecurityManager
- Създадената инстанция трябва да се регистрира (ако услугата RMI Registry не е стартирана – трябва да се стартира)
  – Ръчно: start rmiregistry 2009
  – Програмно: LocateRegistry.createRegistry(2009)

Настройване на политика за сигурност

- Файл с политики за сигурност – виж C:\Program Files\Java\jdk1.6.0_12\jre\lib\security\java.policy
- Задаване на позволения (Permissions) - пример:

```
grant codeBase "file:///C:/myapp/sysadmin/" {
    permission java.io.FilePermission "/resources/abc", "read";
};
```

- Инструмент за създаване и редактиране на файла с политики – policytool

```
policy.all - grant {  permission java.security.AllPermission; };
```

- Опции на Java VM: -Djava.security.policy="policy.all"

Възможност за динамично зареждане на stubs от клиента

```
try {
    ProductController stub;
    ProductController pController = new
ProductControllerImpl();
    System.setSecurityManager(new RMISecurityManager());
    Registry registry = LocateRegistry.createRegistry(1099);
    ProductController stub =
            (ProductController) UnicastRemoteObject.

exportObject(pController, 0);
    registry.rebind("//localhost:1099/ProductController", stub);
} catch (Exception e) {
    e.printStackTrace();
}
```

References

- Software architecture in wikipedia: http://en.wikipedia.org/wiki/Software_architecture
- Java™ RMI Technology documentation page at Oracle® website: http://docs.oracle.com/javase/8/docs/technotes/guides/rmi/
- Oracle® Java™ RMI Simple Tutorial - https://docs.oracle.com/javase/tutorial/rmi/overview.html
- Security Features in Java SE - https://docs.oracle.com/javase/tutorial/security/index.html

Thank’s for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg